H2xS

P3RL

MODuL3

How To Cr34t perl h2xs Module

اولا وفي البداية سوف نعلم ما هو تعريف الموديل او ما يعرف في اللغة الانكليزية بال

Module

التعريف العام للموديل في لغة البيرل:-
الموديل في لغة البيرل هو عبارة هو عبارة عن جزء من كود مكتوب في لغة البيرل من الممكن ان يتم استخدامها بصورة منفردة ومن الممكن ان يتم دمجها مع برنامج اخر مكتوب في لغة البيرل
وتعريف ال h2xs
عملها هي ان تبني

perl extisions from c header files

وهذا الاكستنشن سوف يتضمن دوال و روتينات فرعية التي من شأنها ان تعمل على اعادة القيمة للجمل المعرفة
الان نأتي الى اسماء الموديلات كما سوف نلاحظ في المستقبل ان شاء الله فان الموديل يجب ان يكون له اسم يدعى به ومن الممكن ان يكون اسم الموديل مكون من مقطع واحد مثل

*C0D3

```
spawn
```

هنا في هذا المثال الذي ذكرناه لدينا اسم موديل مكون من مقطع واحد اما عن كيفية تسمية الموديل سوف نأتي على ذكرها في المستقبل
ومن الممكن ان يتم تسمية الموديل بحيث يكون الاسم الذي تمت تسمية الموديل به مكون من مقطعين كما في هذا المثال الاتي

*C0D3

```
Spawn::geek
```

هنا لدينا اسم موديل مكون من مقطعين
وكذلك نفس الحال من الممكن ان يكون لدينا ايضا موديل مكون من 3 مقاطع بشرط انه يتم الفصل بين الجزء الاول و الثاني بهذه الاشارة
::
ونفس الحال مع المقطع الثاني و الثالث
يتم الفصل بينهم بنفس الاشارة

BuIlDiNg P3Rl H2Xs M0DuL3

كيف تتم هذه الفقرة؟؟ تتم من خلال استعمال في البداية وقبل ان نعرف كيف تتم هذه العملية يجب ان نعلم انه لغة البيرل تزودنا بخاصية مفيدة هي تدعى هذه الخاصية ب

(h2xs)

يقوم بقرأة الملفات المكتوبة بامتداد

.h

للمكتبات المكتوبة بلغة سي
وتقوم بصنع هياكل للملفات ذات امتداد ال

.xs

والتي تكون مطلوبة لكي تبني ال

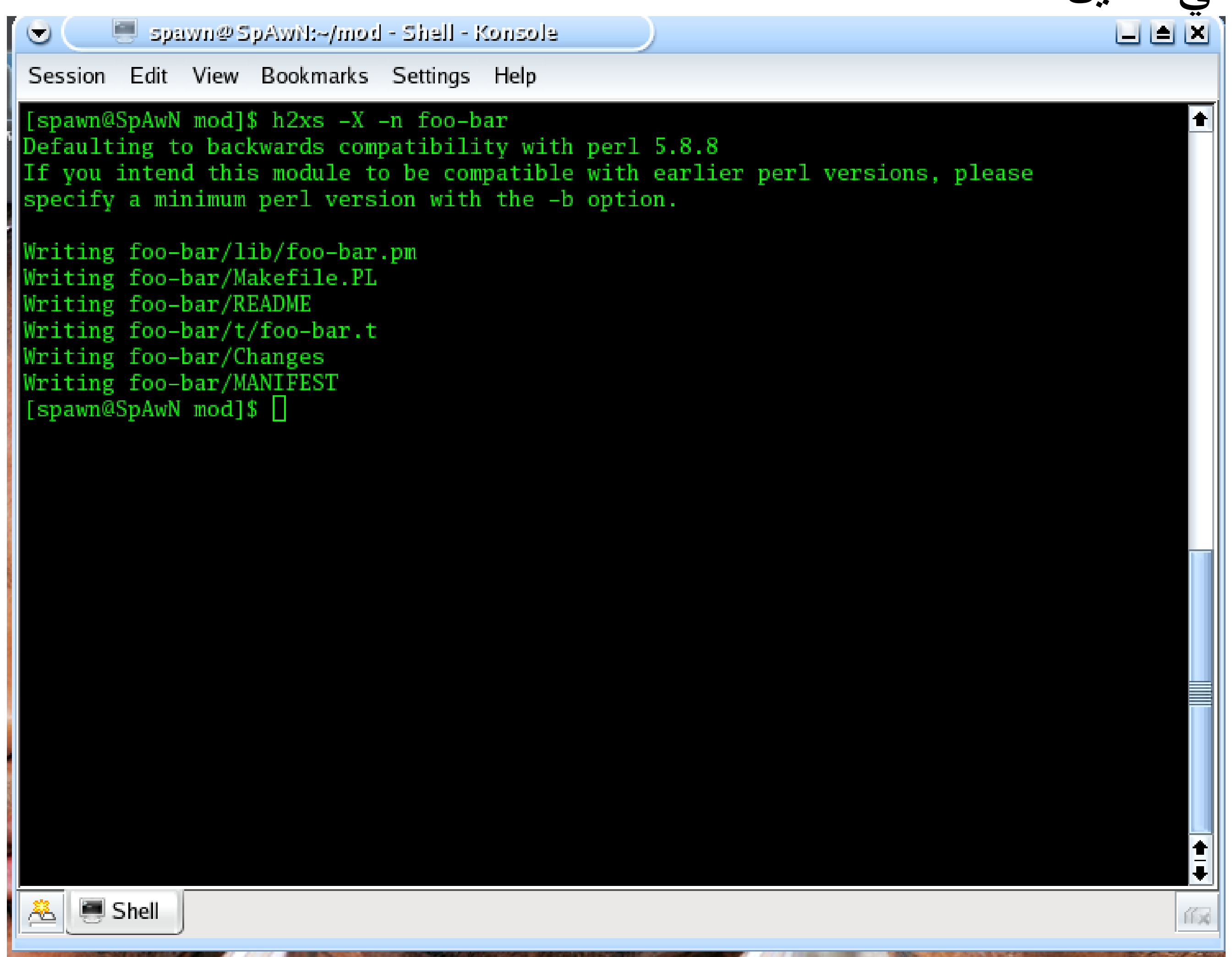
p3rl Module

الان نحن في الجزء المهم من عملية بناء الموديل الخاص بلغة البيرل لكن قبل ما نبدأ نا راح افترض انك على علم بامور الينكس الاساسية التي هي تكوين مجلد في الدليل الاساسي الى اخره من الامور الاساسية التي يتوجب على مستخدم ال لينوكس ان يعرفها الان
اولا افتح الشيل
ومن خلال الامر cd
اذهب الى المجلد الذي تريد ان تتم العملية اي عملية بناء الموديل فيه
ثم بعد ذلك اكتب الامر التالي الذي سوف يكون هو المسئول عن عملية بناء الموديل وهو كما هو موضح في الكود الاتي

*C0D3

```
h2xs -X -n foo-bar
```

الان لو نفذت هذا الكود راح يطلع لك هذه الخطوات او ما يشبهها في الشيل



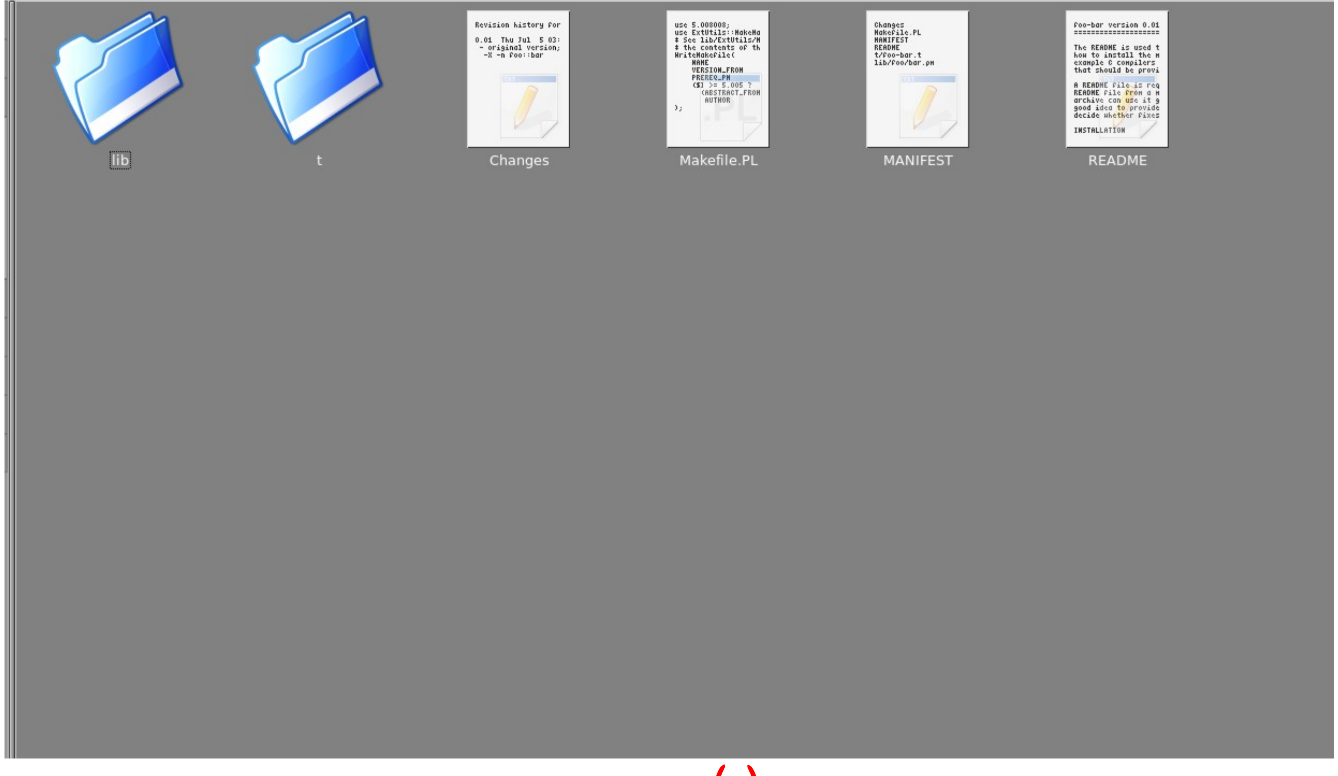
FIGURE(1)

الان لو نذهب الى المكان التي قمنا بتتصيب الموديل فيها سوف نلاحظ ما يلي



FIGURE(2)

الان عندما ذهبنا الى المكان لاحظنا وجود مجلد يحمل اسم الموديل الذي اردنا ان نعمل على تكوينه الان لو قمنا بفتح هذا المجلد سوف نلاحظ انه يتكون من هذه الملفات
مثل التي موجودة في الشكل ادناه



FIGURE(3)

الان نلاحظ انه مكون من مجلدين و اربعة ملفات اي على العموم هو مكون من

6 items

كاملة
الان سوف نعرف ما هو محتوى كل واحدة هذه الملفات و المجلدات ما الذي تعنيه وماذا تحتوي

1.MANIFEST

هذا الملف هو بطبيعة الحالة هو عبارة عن ملف نصي وهذا الملف النصي يحتوي فيما لو قمنا بفتحه يحتوي على 6 أسطر ولو لاحظنا ما هي هذه الأسطر الستة هي عبارة عن اسماء الملفات و المجلدات الموجودة في المجلد الخاص في الموديل اذن هذا الملف النصي يمكن اعتباره اشبه بقائمة تعريف بمحتويات الموديل الذي نحن في داخله

2.README

هذا الملف اقرأني بشكل خاص يحتوي على مقدمة عن الموديلات وعن خطوات فتح الموديل المكتوب في البيرل وكيف. وعلى الرخصة الموضوعة على الموديل واسم الشخص الذي قام بكتابة الموديل ويأخذ اسم الشخص من اسم المستخدم الذي دخل النظام وقام بكتابة الموديل
وعن التاريخ الخ الخ

3.Changes

هذا الملف النصي من اسمه تغييرات يعمل على تسجيل التغييرات التي تتم على الموديل

4.foo-bar.t

هو عبارة عن مجلد فرعي لاختبار الملفات وهذا الملف يحتوي هيكل روتين اختياري للموديل الذي قمت بانشأه كل الذي يقوم بعمله هو اختبار فيما اذا كان الموديل من الممكن ان يتم تحميله الى برنامج اخر مكتوب في لغة البيرل

5.bar.pm

الان نلاحظ وجود مجلد اخر اسمه lib
هذا المجلد الذي اسمه ليب لو قمنا بفتحه سوف تلاحظ انه ايضا يوجد فيه مجلد اخر اسمه

foo

هذا المجلد الاخر لو قمنا بفتحه سوف نلاحظ انه يحتوي ملف اسمه

bar.pm

هذا هو قلب الموديل اي انه هو المديل الذي تم عمله من خلال كل تلك العمليات تكون لدينا هو هذا الموديل الاصلي

6.Makefile.PL

هذا الملف من الملفات المهمة التي تتكون من عمليات بناء الموديل حيث انه حيث انها تحتوي على اسم الموديل و الاسم الشخص الذي كتبه و التاريخ و الفيرجن للموديل الخ هو ملف مهم ايضا لانه جميع التغييرات سوف تتم عليه ما سوف نلاحظ في الصفحات القليلة القادمة ولاتتم على ملف

Makefile.PL

الاخر

Cr34t!ng p3rL Modul3

الى حد الان نحن كنا في عملية بناء موديل البيرل حيث قمنا بكتابة الايعاز الذي يكون مسؤول عن عملية بناء الموديل لكن نحن الان بصدد تكوين موديل البيرل كما هو مكتوب في الاعلى وهذه العملية تكون كما يلي
اولا نتوجه الى المكان الذي وضعنا فيه الموديل القديم من خلال كتابة الامر

cd

في الشيل ثم يتبعه اسم المسار الذي موجود في الموديل
اذا لم تكون تعرف شئ عن هذا الامر الجأ الى الانترنت هناك مئات الدروس التي تعملك عن استعمال هذا الامر
على كل الان نحن في الشيل سوف نكتب ما يلي

*C0D3

```
perl Makefile.PL
```

كما يلي من خلال هذه الصورة

```
[root@SpAWn foo-bar]# perl Makefile.PL
Checking if your kit is complete...
Looks good
Writing Makefile for foo::bar
[root@SpAWn foo-bar]#
```

FIGURE(4)

الان بعض الملاحظات على الصورة اولا عليك ان تدخل الى داخل الموديل وليس ان تدخل الى داخل المجلد الذي يحتوي على الموديل

كما يظن البعض الان لاحظنا ان الفقرة تم بشكل ناجح من خلال الشيل لذا الان سوف نذهب الى موقع الموديل لكي نرى ما الذي تغير عليه من بعد كاتبه هذا الامر
الان بعد ان ذهبنا اليه نلاحظ انه يوجد فيه ملف جديد هو الملف الظاهر في الصورة ادناه



FIGURE(5)

كما نلاحظ هو عبارة عن ملف ضخم نوعا ما يحتوي على عدد كبير من الروتينات الفرعية التي تخصص ما هو عمل الموديل
وكذلك يخبرك باسم الموديل وكذلك دائما يخبرك بأن لاتعمل اي تغييرات على هذا الملف لانه جميع هذه التغييرات سوف يتم فقدانها لذا فهو يخبرك ان تعمل هذه التغييرات التي تريد ان تعملها في ملف ال

make file

الاخر وليس هذا
وغالبا ايضا يحتوي على المسارات التي يحتويها المجلد والتي يكون في داخلها الموديلات لانه هناك بعض الموديلات تكون مكونة من اكثر موديل و النماذج كثيرة في الانترنت
الان نحن انتهينا من عملية التكوين وهذه هي الخطوة الاولى فقط

2.make

هذه هي الخطوة الثانية من عملية تكوين الموديل في لغة البيرل وهي تتم من خلال كتابة هذا الامر في ال

shell

*C0D3

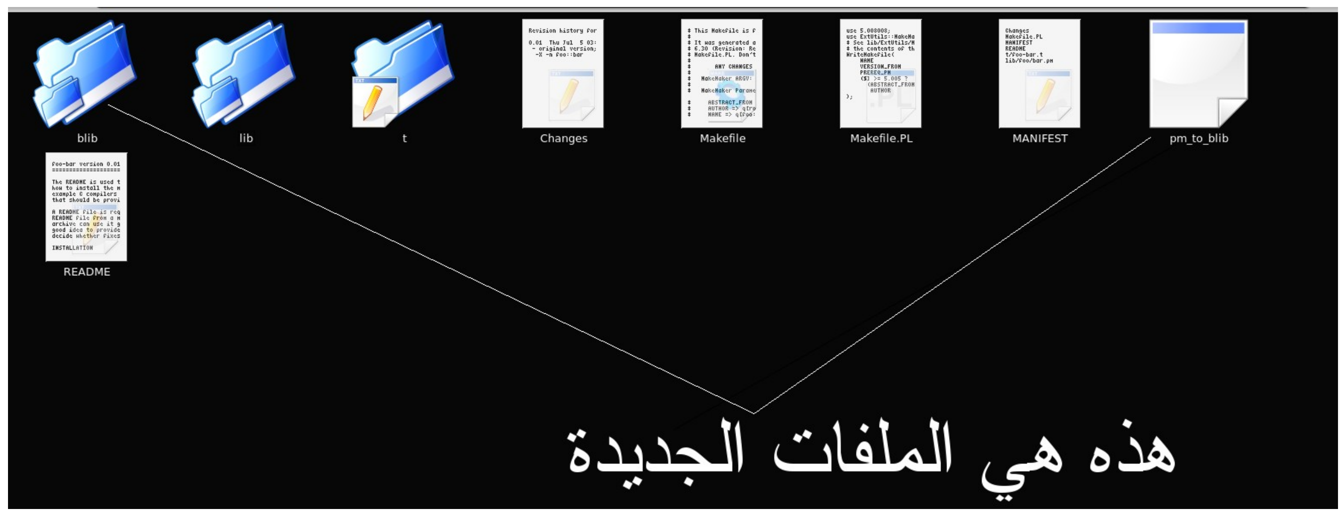
```
make
```

الان نلاحظ انه من بعد ما قمنا بكتابة هذا الامر في الشيل سوف تظهر كتابة تشبه هذه الكتابة او قريبة جدا منها كما في الصورة الاتية

```
[root@SpAWn foo-bar]# make
cp lib/foo/bar.pm blib/lib/foo/bar.pm
AutoSplitting blib/lib/foo/bar.pm (blib/lib/auto/foo/bar)
Manifying blib/man3/foo::bar.3pm
[root@SpAWn foo-bar]#
```

FIGURE(6)

الان لاحظنا ما هو الفرق البرمجي عندما نقوم بكتابة هذا الامر في الشيل سوف نحصل على ما حصلنا عليه في الصورة التي في الاعلى الان سوف نقوم بالتوجه الى مكان الموديل لكي نرى ما هي التغييرات التي طرأت على الموديل عندما قمنا بكتابة هذا الامر او هذا الايعاز
والتي سوف تكون كما في الصورة ادناه التالية



FIGURE(7)

الان بعد ان عرفنا ماهي الملفات الجديدة التي نتجت من تنفيذ الايعاز اصنع و هو كما ذكرنا الخطوة الثانية من عملية تكوين الموديل في لغة البيرل

الملف الاول الذي يحمل الاسم

pm_to_blib

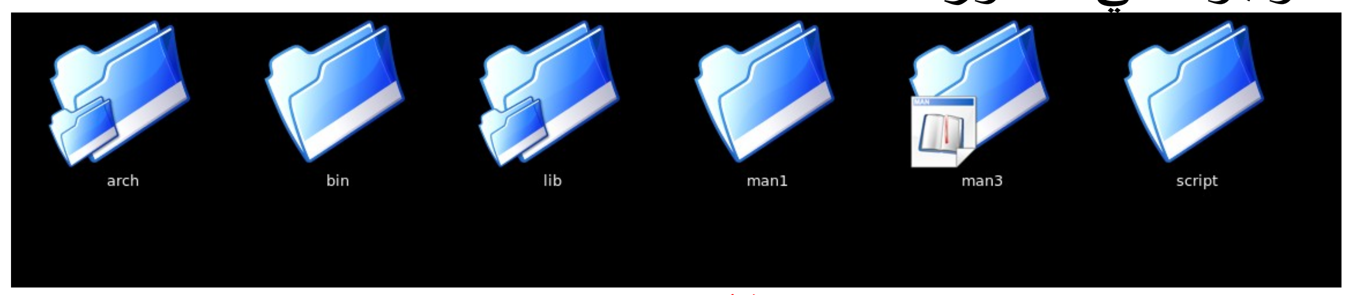
سوف نتركه حاليا لانه ليس مهم جدا في الموديل لانه لو انتبهت الى الحجم الذي يحلمه هذا المف سوف تلاحظ انه يحمل 0 بايت
و هذا الحجم هو غالبا موجود في اغلب الموديلات التي تحتوي على هذا الملف

اما ماذا عن الملف الثاني الذي يحمل الاسم

blib

الان سوف نته الى الموديل و نتفتح هذا الملف لكي نرى من ماذا يتكون هذا الملف وماهي الامور التي تكون في داخله
الان بعد ان فتحنا هذاالملف لاحظنا انه يتكون من الملفات ادناه

الموجودة في الصورة



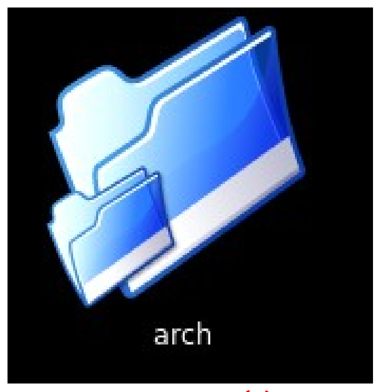
FIGURE(8)

الان نلاحظ انه يتكون من عدد من الملفات التي كما ذكرنا انها نتجت من تنفيذ الامر

make

الان سوف نعلم ماهي محتويات كل مجلد من هذا المجلدات

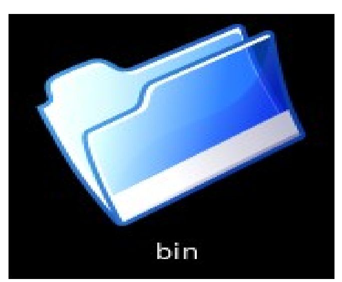
المجلد الاول



FIGURE(9)

هذا هو المجلد الاول ونلاحظ ان هذا المجلد هو عبارة عن مجلد يوجد بداخله عدد من المجلدات والتي تكون غالبا فارغة لذا من ناحية عمل وتكنيك الموديل هذا الملجد لا يحوي على امور قد تكون هامة من ناحية عمل الموديل

المجلد الثاني



FIGURE(10)

هذا هو المجلد الثاني الذي نتج من الامر

make

وهو في اغلب الاحيان ايضا يكون عبارة عن مجلد فارغ هذا عندما نتكلم عن الموديلات القاسية التي نقوم نحن في صنعها اما الموديلات الاخرى التي تمت برمجتها لكي تقوم باعمال معينة فقد يحتوي هذا المجلد على بعض الملفات الثنائية

المجلد الثالث



FIGURE(11)

هذا لمجلد على عكس المجلدات الاخرى هو يحتوي على الملفات اي لا يكون فارغة عندما يتم تكوينه حيث انه لو قمنا بفتح هذا المجلد سوف نلاحظ انه ايضا مكون مجلدين اخوان يكون في داخله ويكونان مثل المجلدان الذي يظهرون في الصورة



FIGURE(12)

نلاحظ ان المجلد الاول لو الذي يحمل الاسم

foo

لو قمنا بفتحه نلاحظ انه يحتوي على نسخة من الموديل
اما الملف الاخر الذي يحمل الاسم

auto

هذا الملجد يحتوي في داخله ايضا على عدد ملجد في داخل الملجد
نلاحظ وجود ملف يدعى

autosplit.ix

حقيقة هذا الملف لم اعرف ما هي اهميته الا انه يعطيك مسار
الموديل الذي يكون في في المجلد الاول الذي ذكرناه قبل قليل
وتجدر الاشارة الى انه هذا الملف

(autosplit.ix)

يكون موجود فقط في الموديلات القياسية التي تنشأ لا لغرض معين
كما هو الحال مع هذا الموديل الذي نعمل عليه اما الموديلات
البرمجية التي يتم برمجتها لاغراض معينه غالبا لاتحتوي على هذا
الملف
كذلك تجدر الاشارة الى انه المف الذي يحمل الاسم

foo

في الموديل الذي قمنا في تكوينه الان غالبا اي في الموديلات البرمجية يكون يحتوي على كل ما يتعلق في الموديلات التابعة للموديل الاساسي لانه هنالك موديلات كبيرة جدا تحتوي على موديلات اساسية و موديلات تابعة

مثل

الموديل الخاص بال

perl tk gui programming

مثل موديل ال

DBI

الخاص ببرمجة قواعد البيانات

مثل موديل ال

CGI

الخاص ببرمجة صفحات الانترنت لذا

انا ذكرت هذه الامثلة كي تلاحظ الفرق وتعرف لماذا يكون هذا الملجد في بعض الموديلات البرمجية ملئ بالكودات بينما الموديل الذي قمنا بأنشأه لا يحتوي على هذا الكم الكبير من الموديلات والروتينات البرمجية الكثيرة

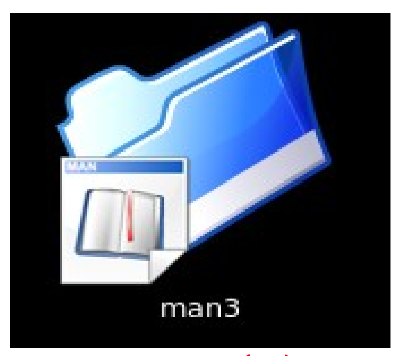
الان نرجع الى المجلد الرابع وهو المجلد الذي ظاهر في الصورة



FIGURE(13)

كما نلاحظ من خلال هذا الملف هو ايضا عبارة عن ملف فارغ لذا لايحتاج الى شرح

المجلد الخامس



FIGURE(14)

هذا لمجلد كما نلاحظ في الصورة
ان الصورة تخبرنا ان هذا المجلد يحتوي في داخله على ملفات اذن الان سوف ندخل الى داخل هذا لملجد لكي نعرف ماهي المكونات التي يتكون منها هذا الملف الان سوف نلاحظ ماهي مكونات هذا الملف من خلال الصورة التي ادنى هذا الشرح



FIGURE(15)

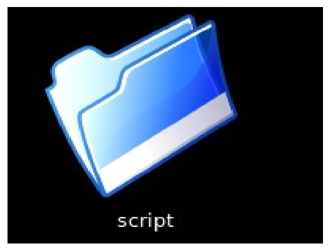
اذن ماهو هذا الملف الذي يحمل اسم الموديل الذي نعمل عليه و هو كما نلاحظ يحمل امتداد غريب
الان لو كنا نحاول ان نفتح هذا الملف سوف نضغط على الزر الايمن من الفأرة ونستعمل الملف برنامج اسمه

kmanpart

هو الذي يكون مسئول عن عرض هذا النوع من ملفات المسادة الذي هذا الامتداد الغريب الذي ذكرناه و الملف الذي يحمل اسم الموديل الذي قمنا بتكوينه ماهو الا صفحة من صفحات المان التي تكون متواجدة في انظمة الينوكس وبالتالي و هذا شئ اكيد كانت لابد ان تتوفر في الموديلات حتى لو كانت موديلات قياسية على كل نحن الان علمنا انه هذه الصفحة هي صفحة خاصة بالمساعدة طبعا تجدر الاشارة الي شئ مهم هو بما انه نحن نعمل على موديلات قياسية اقصد من هذه الكلمة انه لا هدف برمجي منها الا التعلم فان اغلب صفحات المساعدة التي تكون خاصة بهذا النوع من الموديات تكون هي نفس الصفحات الاختلاف يكون فقط من نقطتين اسم المستخدم يختلف و اسم الموديل اما المحتوى البرمجي لهذه الصفحات هو ذاته

ولا يختلف عنه شئ اما الموديلات البرمجية الخاصة باعمال معينه نلاحظ انها مختلفة منعن بعضها البعض وتكون صفحات المساعدة الخاصة بهذا النوع من الموديلات غالبا كثيرة اي اكثر من واحدة

المجلد السادس
هو المجلد الذي يظهر في الصورة ادناه



FIGURE(16)

وهو كما يظهر في الصورة ايضا عبارة عن ملف فارغ لا يحتوي على معلومات

الان نحن انتهينا الخطوة الثانية من عملية تكوين الموديل حيث عملنا ما هي الملفات الجديدة الناتجة وماهي المفات التي نتجت وفائدة هذه الملفات
الان سوف نذهب الى الخطوة الثالثة من عملية تكوين الموديل

3.make install

يعني الان لكي تنفذ الايعاز هذا اكتب فقط هذا في الشيل

*C0D3

```
make install
```

الان عند التوجة الى الشل وكتابة الايعاز هذا سوف تحصل على ما يلي كما في هذه الصورة

```
[root@SpAWn foo-bar]# make install
Installing /usr/lib/perl5/site_perl/5.8.8/foo/bar.pm
Installing /usr/lib/perl5/site_perl/5.8.8/auto/foo/bar/autosplit.ix
Installing /usr/share/man/man3/foo::bar.3pm
Writing /usr/lib/perl5/site_perl/5.8.8/i386-linux-thread-multi/auto/foo/bar/.packlist
Appending installation info to /usr/lib/perl5/5.8.8/i386-linux-thread-multi/perllocal.pod
```

FIGURE(17)

الان نلاحظ انه عند تنفيذ الايعاز هذا نتجت عنه كل هذه الاضافات
الان سوف نتجه ايضا الى مكان الموديل لكي نرى ما هي الاختلافات التي نتجت عندما تم تنفيذ هذا الايعاز
نلاحظ ايضا فقرة خلال تنفيذ هذا الام هو انه تم تنصيب بعض الملفات الخاصة بهذا الامر لكن هذه الملفات التي تم تنصيها كما نلاحظ من خلال الصورة التي في الاعلى لم يتم تنصبيها في مجلد

الموديل لكن نلاحظ انه اغلب الملفات الناتجة من هذا الامر يتم تنصيبها في مجلد ال

/usr

يعني الان خلصنا الى تيجة هي الناتج من تنفيذ هذا الايعاز لا يتم تنفيذه على المكان الموجود فيه الموديل بل على الملجدا ت الخاصة بالنظام

how to make a perl module dist

الان انتهينا من عملية بناء وتكوين الموديل في لغة البيرل وعلمنا ما هي الاوامر الخاصة ببناء وتكوين الموديل والمجلدات التي تنتج من هذه العمليات وما فائدة هذه لمجدات وماذا تحتوي
الان نحن انتهينا من عملية برمجة الموديل وان احببت ان تنشر الموديل الذي قمت بصنعه على الانترنت وانت لاحظت كم كثيرة هي الملفات التي تكون ناتجة عن برمحة الموديل وفي بعض الاحيان لو حتى لم تكن كثيرة وقمت بنسخها قد تنسى بعض الملفات الخاصة بهذا الموديل لذا فأن البيرل اعطت لنا حل هو ايعاز يقوم بعمل هذه لمهمة بدلا عنا
وهو الايعاز التالي

*C0D3

```
make dist
```

الان نلاحظ عند تنفيذ هذا الايعاز في الشيل ما هو الناتج الذي سوف يظهر منه كما في هذه الصورة

```
[root@SpAwN foo-bar]# make dist
rm -rf foo-bar-0.01
/usr/bin/perl "-MExtUtils::Manifest=manicopy,maniread" \
                -e "manicopy(maniread(),'foo-bar-0.01', 'best');
"
mkdir foo-bar-0.01
mkdir foo-bar-0.01/lib
mkdir foo-bar-0.01/lib/foo
mkdir foo-bar-0.01/t
Generating META.yml
tar cvf foo-bar-0.01.tar foo-bar-0.01
foo-bar-0.01/
foo-bar-0.01/README
foo-bar-0.01/MANIFEST
foo-bar-0.01/Changes
foo-bar-0.01/t/
foo-bar-0.01/t/foo-bar.t
foo-bar-0.01/lib/
foo-bar-0.01/lib/foo/
foo-bar-0.01/lib/foo/bar.pm
foo-bar-0.01/META.yml
foo-bar-0.01/Makefile.PL
rm -rf foo-bar-0.01
gzip --best foo-bar-0.01.tar
```

FIGURE(18)

الان نحن نلاحظ ما هو ناتج تنفيذ هذا الامر على الشيل لكن بدلا من ان بقرأة هذه الملفات جميعها لكي نعرف ما هو عمل هذا الايعاز سوف نذهب الى المكان الذي يحتوي على الموديل الذي برمجناه لكي نرى ناتج عمل هذا الايعاز بصورة اسهل و ابسط



FIGURE(19)

وناتج هذا الامر يكون كما هو موضح في الصورة الاتية
الان لاحظنا ان ناتج تنفيذ هذا الايعاز ان يعمل على جعل كل ملفات

الموديل كلها في ملف واحد وهذا الملف يكبس لكي تم الحفاظ على ملفات الموديل ولكي يتم نشرها بصورة جيدة والحفاظ على الملفات من الضياع

ThE StRuCtUr3 of h2xs m4!n L!n3

الان سوف نعود الى البداية اي البداية التي شرحنا فيها كيف يتم تكوين موديل بيرل وذكرنا ان العملية التي تكون مسؤولة عن تكوين الموديل هو السطر التالي

*C0D3

```
h2xs -X -n foo-bar
```

الان لدينا سوال يطرح نفسه بقوة ما هو معني هذا السطر يعني نحن عملنا ان الحروف الحروف الاربع الاولى من هذا السطر تكون هي المسؤولة عن تكوين الموديل وكلمة

*C0D3

```
foo-bar
```

هي التي تحمل اسم الموديل الذي سوف نعمل عليه اذن الان علمنا ما هي فائدة هذه الفقرات الان سوف نتعلم الاحرف الموجودة في السطر الذي من خلاله نقوم ببرمجة الموديل
يعني الان لدينا سؤال هو لماذا قمنا باستخدام هذه الاحرف ندما قمنا ببناء الموديل حيث كان السطر الي تم استخدامه هو السطر التالي

*C0D3

```
h2xs -X -n foo-bar
```

الان نود ان نعلم لماذا قمنا باستخدام الحرف

X& n

وكان لحرف اكس في الحالة الكبيرة من الاحرف وكان الحرف أن في الحالة الصغيرة الان سوف نتعلم ما هي فائدة هذه الاحرف

وماهو الفرق بين الحالة الكبيرة و الحالة الصغيرة حتى بين نفس الاحرف

حالات الاحرف

1.

-A

الان سوف ناخذ هذه الحالة التالية في الكود الاتي سوف نقارن الفرق بين هذه الحالة التي سوف ناخذها و الحالة التي كنا نعم عليها في الماضي

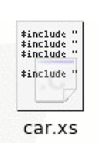
*C0D3

```
h2xs -A -n foo-car
```

هذا الامر الذي يكون مسؤول حاليا عن تكوين موديل جديد لكن هذا الامر لكي نلاحظ عمله ننفذه في الشيل الان سوف لن اعرض ما هو ناتج الامر على الشيل لكي لا يطول الشرح الان مباشرة سوف نذهب الى المكان التي يوجد فيها الموديل لكي نلاحظ ما هو الفرق الان هو هذا ناتج تنفيذ الايعاز كما هو واضح من خلال الصورة التالية

















FIGURE(20)

الان نلاحظ هل هنالك فرق ؟؟

من ناحية هل يوجد فرق ام لا اكيد يوجد فرق ولكن ليس هذا الفرق ما هو الذي يهمنا الان سوف نتجه الى المكان الفعلي الذي وجد به

الموديل ونفتحه ونلاحظ انه لايحتوي على هذه الفقرة الواضحة في الصورة
الان عندما فتحنا ملف الموديل لفعلي لهذا الموديل الذي برمجناه لاحظنا انه مكون مما يلي

```
package foo::car;

use 5.008008;
use strict;
use warnings;

require Exporter;

our @ISA = qw(Exporter);

# Items to export into callers namespace by default. Note: do not export
# names by default without a very good reason. Use EXPORT_OK instead.
# Do not simply export all your public functions/methods/constants.

# This allows declaration       use foo::car ':all';
# If you do not need this, moving things directly into @EXPORT or @EXPORT_OK
# will save memory.
our %EXPORT_TAGS = ( 'all' => [ qw(

) ] );

our @EXPORT_OK = ( @{ $EXPORT_TAGS{'all'} } );

our @EXPORT = qw(

);

our $VERSION = '0.01';

require XSLoader;
XSLoader::load('foo::car', $VERSION);

# Preloaded methods go here.

1;
__END__
# Below is stub documentation for your module. You'd better edit it!

=head1 NAME
```

FIGURE(21)

*N0T3

عندما اقول الموديل الفعلي يعني من هذه الكلمة الملف الذي ينتهي بالامتداد

.pm

لذا ارجو الانتباه الى هذه النقطة
على كل الان بعد ن لاحظنا هذا المحتوي للموديل الفعلي نلاحظ انه لا يحتوي على اي شئ غريب يدعو الى الانتباه او يجلب الشك لا الان سوف الصورة الخاصة بالموديل الفعلي الذي كنا نعمل عليه في بداية الكتاب لكي نلاحظ الفرق

```
package foo::bar;

use 5.008008;
use strict;
use warnings;

require Exporter;
use AutoLoader qw(AUTOLOAD);

our @ISA = qw(Exporter);

# Items to export into callers namespace by default. Note: do not export
# names by default without a very good reason. Use EXPORT_OK instead.
# Do not simply export all your public functions/methods/constants.

# This allows declaration       use foo::bar ':all';
# If you do not need this, moving things directly into @EXPORT or @EXPORT_OK
# will save memory.
our %EXPORT_TAGS = ( 'all' => [ qw(

) ] );

our @EXPORT_OK = ( @{ $EXPORT_TAGS{'all'} } );

our @EXPORT = qw(

);
```


FIGURE(22)

اذن فعلا هذا هو الفرق الموجود بين الموديل القديم و هو الذي يحمل الصورة التي فوق هذا الكلام مباشرة و الصورة التي في الصفحة السابقة

summary

اذن الخلاصة من استعمال الحرف

A

في الحالة الكبيرة هو انه يعمل على الغاء الفقرة الخاصة باستعمال ال

AutoLoader

ملاحظة اخرى ايضا هي انه المجلدات التي لم من مألوفة في الموديل القديم اي كنا نعمل عليه سوف نشرحها ما هي لذا لا تقلق من هذه الناحية

2.

-C

ان الغرض من استعال الحرف

C

في الحالة الكبيرة هو من اجل ان يتم التخلص من الملف الذي يحمل نالاسم

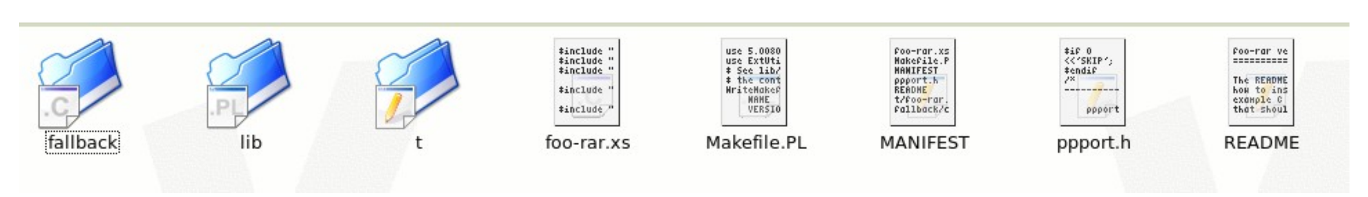
Changes

و الذي كما ذكرنا من خلال الشرح السابق هو انه يعمل اي الملف الخاص بالتغييرات انه يعمل على تسجيل التغييرات الخاص التي تتم على عمل الموديل الان سوف نلاحظ ما اذا كان عمل هذا الامر صحيح

*C0D3

```
h2xs -C -n foo-rar
```

الان سو نلاحظ عمل هذا الايعاز فيما اذا كان صحيحا ام لا من خلال هذه الصورة



FIGURE(23)

الان نلاحظ انه الكلام كان صحيح ومن الممكن ان يتم الغاء الملف الخاص بتسجيل التغييرات

3.

-O

هذه الحالة تكمن فائدتها انها تعمل على القبول بعمليات الكتابة الفوقية او ما يعرف بالغة الانكليزية بمصطلح ال

over write

حيث انه لو قمنا بكتابه هذا الايعاز

*C0D3

```
h2xs -C -n foo::nar
```

الان لو قمنا بتنفيذ هذا السطر و يتم برمجة الموديل بشكل عادي لكن لو اردنا ان نعمل على برمجة اموديل من جديد فأننا سوف لن نحصل على على الامكانية لعمل هذه للخطوة لانها سوف يتم اعتبارها عملية كتابة فوقية لذا فأن استعمال الحرف

O

في الحالة الكبيرة فأنها سوف تسح لنا بعمليات الكتابة الفوقية ولو مئة مرة على التوالي هذه هي الفائدة من هذا الحرف في حالته الكبيرة

4.

-X

هذا الحرف في حالته الكبيرة يعتبر من اهم الحالات التي تكون مصاحبة في عمليات برمجة الموديلات في لغة البيرل حيث بصورة عامة يتم تقسيم الموديلات في لغة البيرل الى نوعين من الموديلات وهذه النوعين يتم استعاملها بكثرة لكن ماهو اساس هذا التقسيم ؟؟ يكون اساس على ان الموديل يحتوي او لا يحتوي مكتبات خاصة

بلغة سي او لا و هذا ما لاحظناه على الموديلات التي تم عملها
و الان سوف نلاحظ لفرق من خلال هذه الكودات

*C0D3

```
h2xs -X -n foo::bar
```

ناتج تنفيذ هذا الايعاز سبق وان رأينا ما هي نتيجته لا انا لن اضع الصورة الخاصة بهذا الايعاز
هذا النوع الاول من الموديلات الذي ذكرنا انه لا يكون اتصال بينه وبين مكتبات لغة السي
النوع الثاني من الموديلات هي الموديلات التي يكون لها اتصال مع مكتبات لغة السي
ويكون النوع الثاني كما في هذا الكود

*C0D3

```
h2xs -A -n foo::mar
```

هذا السطر البرمجي فيما لو اذا تم تنفيذه سوف ينتج عنه هذا الموديل الذي يظهر في الصورة



FIGURE(24)

الان عند تنفيذ هذا السطر البرمجي نلاحظ فعلا يوجد ملفات خاصة

في لغة السي ومن هذا الكلام نتأكد ونقول انه الموديلات في لغة البيرل تكون على نوعين نوع له علاقة في مكتبات السي مثل جميع الموديلات التي لا يتم استخدام الحرف اكس في الحالة الكبيرة و الموديلات التي يتم استعمال الاحرف الاخرى التي ذكرناها كلها يكون لها علاقة بمكتبات السي

5.

-X

هذا الحرف ذكرنا انه يعمل على الغاء الاجزاء الخاصة بمكتبات لة السي ولن اشرحا كثر لانه شرحنا عليها كثيرا
الان انتهينا من شرح الاحرف الكبيرة التي تكون موجودة مع الاسطر البرمجية الخاصة ببرمجة الموديلات

*N0T3

يتم وضع الحرف

n

في الحالة الصغيرة من اجل يتم تخصيص الاسم بشكل جيد ومتوافق وعن الحروف الصغيرة الباقية سوف يتم يتن دمجها في الاعداد القادمة من الكتب عندما يتم مناقشة ودراسة البرمجة المتقدمة للموديلات المكتوبة بلغة البيرل


P3rL DebuggEr

P3rL DebuggEr

The perl Debugger

الان وفي هذا العالم الذي نعيش اغلب الناس التي تكتب البرامج توقع انه
البرنامج الذي تكتبه هو الافضل من نوعه ولكن في الواقع حتى افضل
المبرمجين يعني في لغة البيرل هذه او في لغات اخرى فأنهم عندما يقومون
بكتابه البرامج فأن هذه البرامج بعض الاحيان تكون خاطئة فقد ينسى بعض
الامور الاساسية التي تكون ضرورية لتنفيذ البرنامج وعندما يقوم المبرمج بعمل
فحص للبرنامج الذي كتبه فقد يجوز في بعض الاحيان لن يقدر لعى ان يكتشف
ما هو الخلل الذي ارتكبه لذا دعت الحاجة الى ظهور الديبغر

what is the perl debugger

هو عبارة عن تطبيق يقوم بتتبع البرنامج بينما هو ينفذ وفي هذه الحالة يتم
اختصار الوقت واظهار البكس الموجودة في برنامجك مع البيرل ديبكر يمكنك
ان توقف البرنامج في نقاط او مواقف انت تحددها ويمكنك ان تطبع المتغيرات
او تعرف محتوى هذه المتغيرات

why using the perl debugger

توجد عدة طرق تكون متوفرة لمعرفة مالخطأ الذي يجري في داخل برنامج
البيرل ومن هذه الطرق اعادة قرأة البرنامج مرة اخرى على سبيل المثال او
اعادة برمجة البرنامج مع مبرمج اخر
وفي بعض الاحيان تكون الاخطأء في عدم معرفة اين يتم وضع جملة الطباعة
لانه جملة الطباعة عندما يتم وضعها في مواقع استراتيجية من البرنامج من
الممكن ان تكشف عن وجود بعض الاخطأء التي كانت السبب في وجود الخلل
في تنفيذ البرنامج
ولكن كما ذكرنا في بعض الاحيان يجب عليك ان تعرف انه جملة الطباعة ليست
لوحدها كافية يعني ان الذي اقصده انه جملة الطباعة الموجودة في البرنامج
ليست كافية لذا في هذه الحالة
يجب عليك ان تعمل على معاينة المشكلة التي تعاني منها يعني مثلا في هذا
الكود الذي سوف نأخذه الان عليك ان تعرف ما يلي

*C0D3

```
$a=4;
$b=9;
$c=$a*$b;
print $c;
```

يعني الان لدينا هذا الكود البسيط نلاحظ عند تنفيذ هذا الكود انه الناتج من التنفيذ هو 36 اذا الخلل هنا غير موجود وان جملة الطباعة موجودة في مكانها الصحيح وعملت على تنفيذ البرنامج

لكن الان نحن بصدد تنفيذ برنامج سوف يكون فيه خلل و الخلل من جملة الطباعة

*C0D3

```
$a=4;
$b=9;
$c=$a*$b;
print c;
```

الان الخلل الموجود في هذا البرنامج هو عدم وجود علامة

$

التي تدل على ان الحرف سي هو عبارة عن متغير وليس حرف نصي هذا الذي كنت اقصده

*N0T3

ملاحظة هامة جدا هو بعثرة جملة الطباعة في مختلف انحاء البرنامج هو ليس بالامر الجيد لذا حاول بشتى الطرق ان يكون برنامجك مقتضب وسهل وقابل للفهم

The Guts Of The Perl Debugger

ان كثير من الناس و المبرمجين على نوعيهم الخبراء و المستجدين في الغالب لا يحبون استعمال الديبكر سواء في لغة البيرل او في لغات اخرى لا نهم كما
يظنون وهذا الظن مسبقا هو خاطئ هو ان الديبكر صعب وغير قابل للفهم بل انا على العكس اظن انه سهل ويكون اليد اليمنى لك اذا كنت تعمل على تنفيذبرنامج وكان عندك خطأ وما تعرف شو هو الخطأ الموجود الذي يواجهك
الخطوات التي يتم من خلالها عمل فحص للبرنامج
الطريقة الاولى
هي فحص البرنامج من خلال هذه الطريقة او من خلال هذا المفتاح

*C0D3

```
perl -c
```

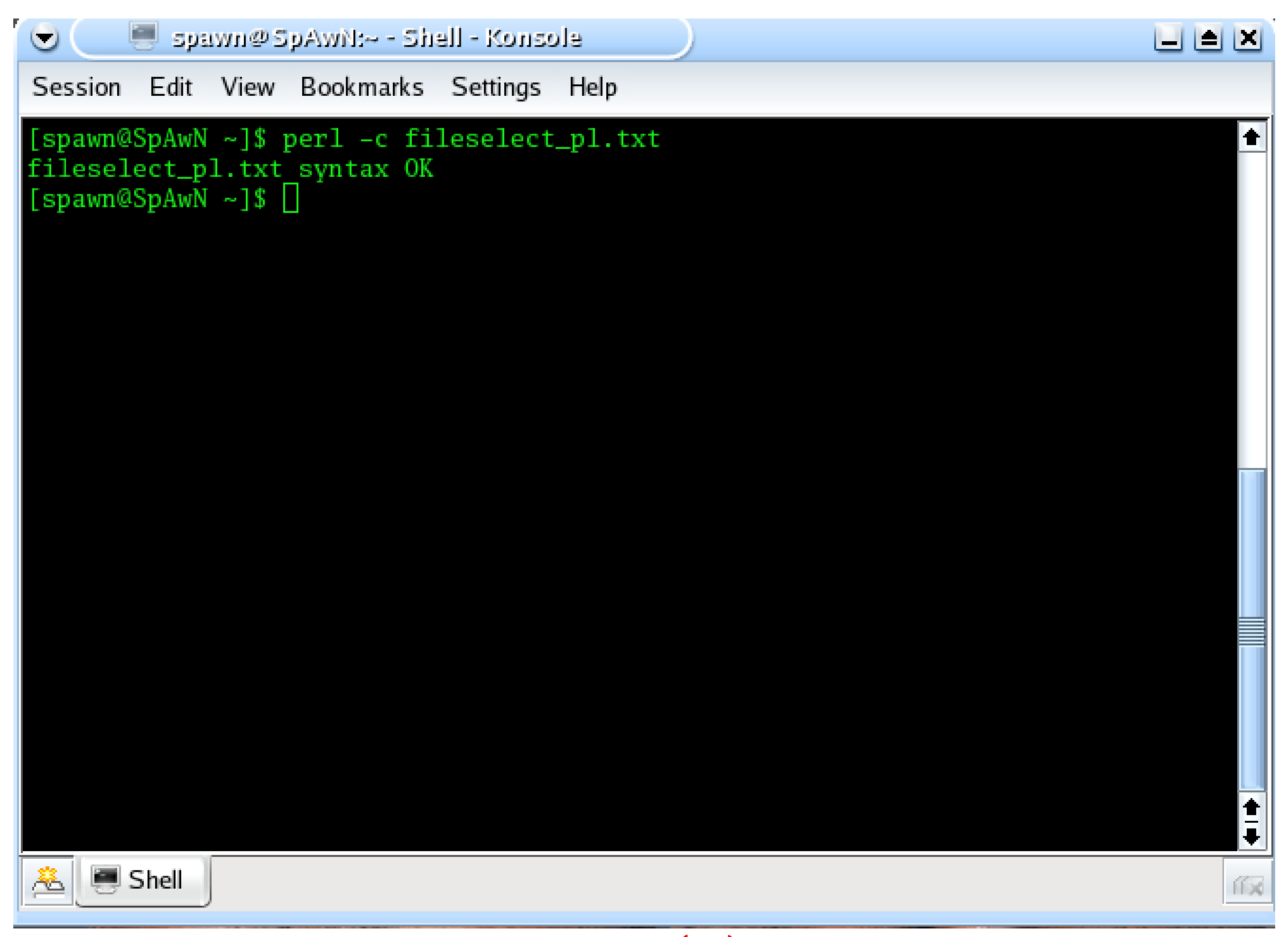
قبل ما ان اعمل شرح على هذا المفتاح ملاحظة هي انه حرف السي الموجود في هذا المفتاح هو لا يعني كلمة

check

بمعني فحص بل يعني حرف السي الموجود هنا

compile

اي ترجم او ترجمة
ويتم استخدام هذا المفتاح كما يلي من خلال هذه الصورة



FIGURE(25)

و اذا كان ال

syntax

الخاص بالبرنامج او الخاص بالمقطع البرمجي صحيح وسف نحصل على النتيجة الظاهرة اعلاه في الصورة
كلمة

fileselect_pl.txt

هي عبارة عن اسم البرنامج الذي نريد ان نعمل كوبايل له

use strict

هو استخدام ما يسمى بال

strict

او ما يعني باستخدام الوضع الصارم او الوضع الدقيق

لاحظ انه عندما يكون برنامجك كامل فأن وضعه في ال
حالة ال

strict compliant

لإان هذا يدل على خطأ كبير خاصة عندما يكون البرنامج ليس من برمجتك او
تكون قد نقلته او الى اخره من طرق لانقل لانه هذا قد يودي الى حدوث اخطاء
كبيرة في البرنامج
لانه من الممكن انه قد تم استخدام بعض المتغيرات في اماكن متعددة من
البرنامج لانه هناك مثلا على سببيل المثال برامج شاهدتها فيها بعض المتغيرات
قد دخلت في اماكن عددين من البرنامج لذا فأن نفعيل هذا الوضع كما ذكرت
خطأ كبير
من الاشياء التي تدل على لغة البيرل هي لغة برمجة قوية هو انه هذا الوضع هو
وضع اختياري يعني يمكن استعماله وقت ما تريد ويمكنك ان لا تستعمله ايضا
الان ما هو الفرق بين هذا الوضع و الوضع العادي
الان لو كان لدينا هذا البرنامج الموجود ادناه

*C0D3

```
@a=("perl","securitygurus");
print @a;
```

الان لو نقوم بتنفيذ هذا الكود سوف نحصل على القيمتان الموجودتان داخل هذه
المصفوفة هذا في الوضع العادي
اما في وضع ال

strict mode

سوف ناخذ هذا الكود

*C0D3

```
use strict;
@a=("perl","securitygurus");
print @a;
```

الان لو نفذنا هذا الكود سوف نحصل على قائمة في الاخطاء تدلنا على انه هذا
الكود يحتوي على عدد من الاخطاء البرمجية لكن لو قمنا بصياغة هذا الكود
بطريقة اخرى سوف يكون بالامكان تلافي هذا الخطأ

*C0D3

```
use strict;
my @a=("perl","securitygurus");
print @a;
```

الان لو قمنا بتنفيذ هذا الكود سوف نحصل على ناتج التنفيذ اذن
الشئ الذي نعلمه
الان ناخذ تعريف ما هو ال

strict

التعريف الرسمي لها هو
ان ال

strict pragma

يعمل على انكار او نفي ال

soft reference

وهو يعمل على التاكد من انه كل المتغيرات التي تم ادخالها يتم تعريفها قبل
الاستخدام
ويعمل على انكار ال

barewords

ما عد ال

barewords

التي يتم استعمالها في الروتينات الفرعية
ها هو التعريف الرسمي للوضع الصارم او الوضع الدقيق
ملاحظة
كما علمنا ان الوضع الدقيق او الصارم هو وضع اختياري اي انه من الممكن ان
يتم استخدامه ومن الممكن ان لا يتم استخدامه ولكن هنالك فقرة اخرى هي انه
يتم تعطيله من خلال كود برمجي يعمل على هذا التعطيل

*C0D3

```
use strict;
no strict;
@a=("securitygurus");
print @a;
```

نلاحظ انه الان اذا قمنا بطباعة المصفوفة هذه سوف نحصل على القيم الموجود فيها لانه يوجد لدينا الان الكود الذي يعمل على تعطيل الوضع الصارم لو قمنا بالغأه سوف نحصل على خطأ

warnings

او ما يعرف في العربي بالتحذيرات عندما يتم تفعيل الوضع التحذيري يعمل على جعل البير لتعمل على اصدار رسائل تحذيرية بشأن الكودات المشبوهة التي يتم الاشتباه بها انها تحتوي على بعض الاخطأء البرمجية
يتم تشغيل الوضع التحذيري في لغة البيرل من خلال هذا المفتاح كما في هذا الكود

*C0D3

```
perl -w
```

هذا هو المفتاح الذي يعمل على تشغيل الوضع التحذيري في لغة البيرل
مثلا لو كان لديك هذا الكود وقمت بتنفيذه وكان الوضع التحذيري مفعل

*C0D3

```
print $ARGV[0];
```

فانك سوف تحصل على هذه الرسالة

*C0D3

```
Use of uninitialized value in print at - line 1.
```

بينما لو كنت في الوضع العادي يعني وكنا الوضع التحذيري غير مفعل وكتبت هذا الكود وقمت بتنفيذه فانك كنت لن تحصل على اي خطأ الان لنجرب كود اخر

*C0D3

```
print $a;
```

الان ايضا لو نفذت هذا الكود و الوضع التحذيري مفعل كنت راح
تحصل على سطرين من الاخطاء هما

*C0D3

```
Name "main::a" used only once: possible typo at - line 1.
Use of uninitialized value in print at - line
```

وهكذا الحال مع هذا الوضع
هنالك طريقة اخرى لكي يتم استخدام الطريقة التجذيرية او الوضع التحذيري
وذلك يتم كما ما هو موجود في هذا الكود

*C0D3

```
use warnings;
print $a;
```

هذه الطريقة هي نفس الطريقة السابقة التي قمنا باستخدامها وعند تنفيذ هذا
الكود فأننا سوف نحصل على نفس الناتج وعلى نفس الاخطاء التي حصلنا عليها
من الطريقة الاولى من الوضع التحذيري
اما اذا كنت تريد ان عمل على اغلاقها
ففي هذه الحالة هناك عدد من الطرق التي قد تساعدك على اغلاق الوضع
التحذيري
اولا
اذا كنت تريد ان تعمل على اغلاقها بشكل موقعي ففي هذه الطريقة من الممكن
ان يتم استخدام هذا المفتاح البرمجي

*C0D3

```
no warnings;
```

هذا المفتاح يمكن القول عليه انه النفي او النقيض لحالة الوضع التحذيري الاول
اما الطريقة الثانية التي من الممكن ان تعمل على ايقاف الوضع التحذيري هي
عن طريق استخدام هذا المفتاح

*C0D3

```
perl -X
```

هذا المفتاح يعمل على ايقاف الوضع التحذيري
ملاحظة هنا حرف الاكس هو في الحالة الكبيرة لذا يرجى الانتباه
وهذا المفتاح يعمل على ايقاف الوصع التحذيري حتى لو كان البرنامج مكون مما يلي

*C0D3

```
use warnings;
print $a;
```

اي حتى لو كان البرنامج يحتوي على السطر البرمجي الذي يعمل على تفعيل الوضع التحذيري فأن هذا المفتاح يعمل على تعطليه

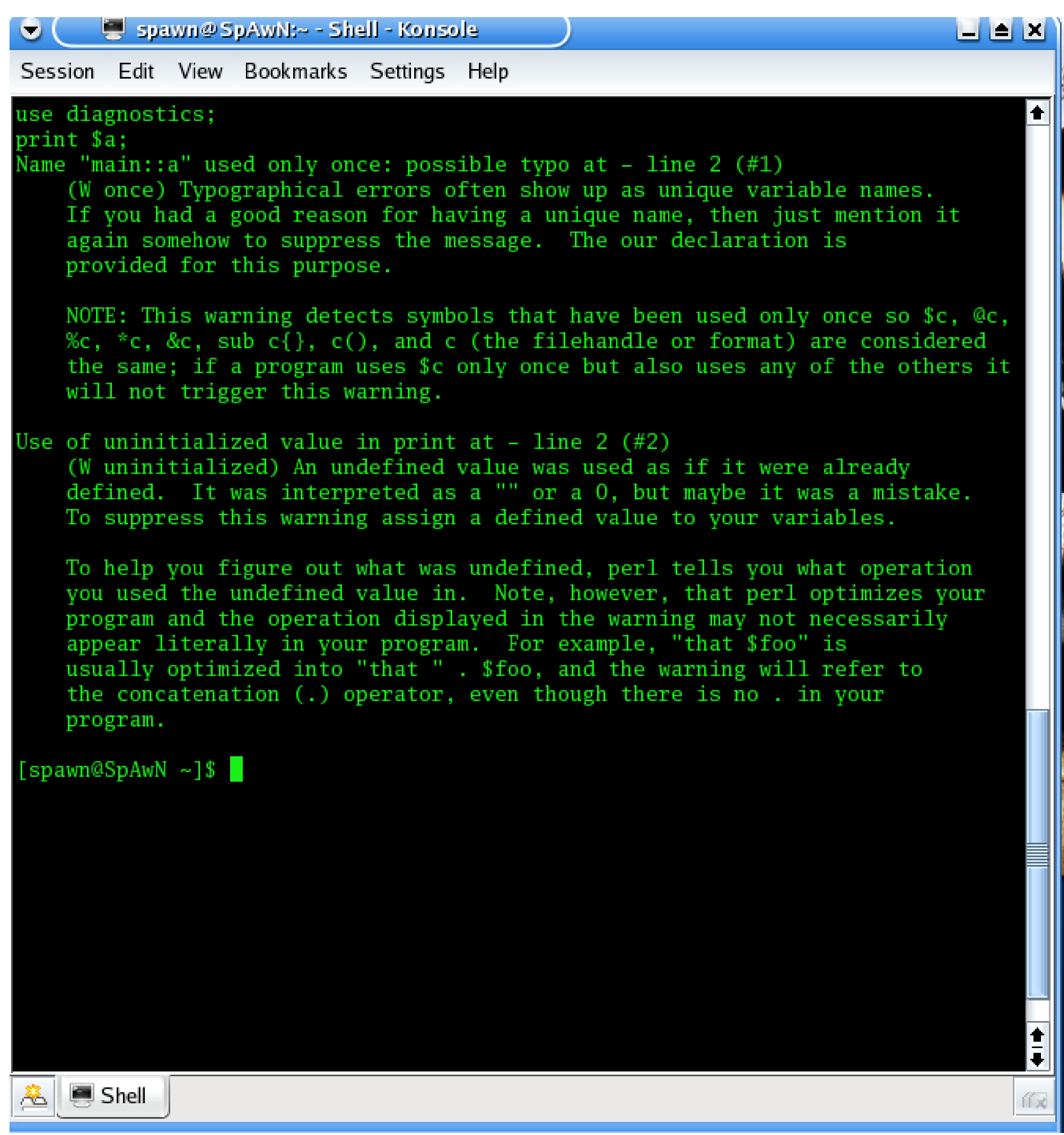
Diagnostics

او ما يعرف في اللغة العربية على انه الوضع التشخيصي اذا الى حد الان كانت التقارير عن الاخطاء التي تظهر لديك ليست مقنعة لك او لست مقتنع بمدى فعالية هذه التقارير يمكنك ان تقوم باستخدام هذا الوضع كما في هذا الكود

*C0D3

```
use diagnostics;
print $a;
```

الان لو قمت بتنفيذ هذا الكود سوف تحصل على صفحة كاملة تشرح لك ما هي عبارة الاخطاء التي لديك كما في هذه الصورة



FIGURE(26)

الان نلاحظ انه هذا الوضع يوضح لنا مقدار كبير من المعلومات عن الخطأ الذي قمنا بارتكابه طبعا تجدر الاشارة الى انه كل حالة تختلف في عرض الاخطاء التي يقوم هذا الوضع بعرضها يعني هنا لدينا صفحةكاملة تقريبا عن الاخطاء التي يعرضها الوضع التشخيصي عندما قمنا باستخدام متغير لو لو غيرنا النتغير

الى مصفوفة سوف تتغير الوضعية ونفس الحال مع البرامج الاكبر و الاكثر تعقيدا
هنالك ملاحظة هامة كان لابد من الاشارة اليها هي انه عندما يكون المفسر الذي تستخدمه حديث او قديم قد لا يظه لك نفس الاخطاء التي موجودة في الصورة لكن سيظهر لك شئ يشبهها بنسبة 90 بالمئة

Taint M0d3

اذا كنت من الاشخاص الذي مهووسين بكاملية البرامج فأن هذا النمط سوف يكون مساعد جدا لك حيث انه يعمل على معاملة كل متغير من خارج البرانمج على انه متغير مشكوك فيه وانه متغير قد يشكل خطر في البرنامج الذي تعمل عليه
وانه سوف يمل على عدم تنفيذ البرامج من اماكن غير معروفة من النظام ومن الممكن ان يتم تفعيل هذا النمط من خلال هذا الكود الاتي

*C0D3

```
perl -T
```

الان سوف نعمل على تجريب حركة صغيرة هذه الحركة من خلالها سوف نتعلم من خلال هذه الحركة سوف نتعلم انه هذه الحركة سوف لن تسمح لنا بالقيام بعدد من الحركات التي نقوم بها فيما لو اذا كان هذا النمط غير مفعل من خلال الكود الاتي
اولا بل ما نأتي اى هذا الكود الكل يعرف ما هو الامر

echo

واذا كنت لاتعرف ما هو عمل هذا الامر عليك ان تستعرض صفحة المساعدة الان عليك ان تلاحظ الصورة التي سوف يتم عرضها لكي يتم معرفة ماالذي يتم عندما يتم تفغيل هذا الوضع ومالذي يحدث عندما لا يتم تنفيذه

```
[spawn@SpAwN ~]$ perl -T
system "/bin/echo Spawn";
Insecure $ENV{PATH} while running with -T switch at - line 1.
```

FIGURE(27)

هذا الذي يحدث عندما يكون هذا النمط مفعل

الان مالذي سوف يحدث اذا لم يكون هذا النمط مفعل
هذا سوف نلاحظه في الصورة الاتية

```
[spawn@SpAwN ~]$ perl
system "/bin/echo Spawn";
Spawn
```

FIGURE(28)

وكالعادة اذا كنت من محبي الشرح المفصل و الديقيق عن الاخطاء التي تكون موجودة في البرنامج الذي تعمل عليه من الممكن ان يتم الحصول على ملعومات اكثر وذلك من خلال الامر او المفتاح الاتي

*C0D3

```
perl -t
```

تماما كما يوجد في الصورة الاتية

```
[spawn@SpAwN ~]$ perl -t
system "/bin/echo Spawn";
Insecure $ENV{PATH} while running with -t switch at - line 1.
Insecure dependency in system while running with -t switch at -
line 1.
Spawn
```

FIGURE(29)

ولكن هذا النمط يحتوي على نقطة اختلاف لو دققتم بها سوف تعلمون ماهي ركزوا قليلا

*N0T3

نلاحظ من خلال هذه الصور ان الكودات هي نفس الكودات ولكن الذي اختلف ان هو نمط تشغيل البرنامج او ما يمكن ان نطلق علية البيئة الامنية الخاصة بالبرنامج لكي يتم التأكد من ان البرنامج امين ولا يحتوي على شئ خطر

starting perl debugging session

الان لكي نبدأ جلسة العمل على الديبكر او ما يطلق عليه البعض على انه
مصحح الاخطاء على كل الان نحن سوف نعمل على اعطاء قيمة هذه القيمة
سوف تكون قيمة بسيطة وغير موذية لانها سوف تكون القيمة التي سوف نعمل
عليها التصحيح وال
evaluation
سوف نكتب التالي في سطر الاوامر

*C0D3

```
perl -d -e xx
```

d=debuger
e=make evaluation
xx=the name of your program

الان عندما يتم الدخول الى مصحح الاخطاء فأنه لن يعمل على تنفيذ البرنامج
خظوة خطوة لانه هذه عملية مزعجة وطويلة لذا فأنه سوف يعمل على تنفيذ عند
الخطوات التنفيذية فقط
كما الاتي
لو كان لديك البرنامج الاتي

*C0D3

```
$a="spawn";
$b="geek";
print $a;
print "\n";
print $b;
```

فأن عملية تنفيذ هذا البرنامج سوف تكون ما يلي
في الصورة الاتية

```
   DB<1> $a="spawn";
$b="geek";
print $a;
print "\n";
print $b;



   DB<2>
   DB<3> spawn
   DB<4>

   DB<5> geek
   DB<6>
```

FIGURE(30)

الان شرح عن هذه الصورة اولا مصطلح ال

*C0D3

```
DB<2>
```

هذا المصطلح الموجود في بداية السطر تعني هذه

*C0D3

```
Debugger line
```

يعني رقم السطر المتسلسل الموجود في مصحح الاخطاء
كذلك نلاحظ في هذه الصورة ما يلي ان السطر الثاني لا يحتوي على كلمة او
تنفيذ او اي شئ اي انه فارغ تماما لكن السطر الثالث يحوي على كلمة

*C0D3

```
spawn
```

والسطر الرابع ايضا لايحتوي على شئ بينما السطر الذي بعده يحتوي كلمة

*C0D3

```
geek
```

الان سوف نتناول شرح هذه العملية
سبب فراغ السطر الثاني هو انه لا يحتوي على جملة تنفيذية ونحن عند بداية الشرح ذكرنا ان مصحح الاخطاء لا ينفذ الاسطر التنفيذية بينما السطر الذي بعده يحتوي على كلمة اطبع المتغير لذا قام بعملية الطباعة بينما السطر الذي بعده هو حقيقة ليس سطر فارغ بل هو سطر تنفيذي لانه لو رجعت الى البرنامج لاحظت انه هذا السطر يحتوي على جملة ال

*C0D3

```
print "\n";
```

وهذا السطر البرمجي وظيفته هي انه يقوم بطباعة الاسطر الفارغة
اما السطر الذي بعد هذا السطر يكون سطر تنفيذي لذا فأنه يحمل نتيجة ويقوم مصحح الاخطاء بتنفيذ هذا الامر او السطر البرمجي
وعلى هذا الاساس يعمل مصحح الاخطاء

ملاحظة هامة
عن مصحح الاخطاء هو انه يمكن ان يتم استدعاء صفحات المساعدة الخاصة به من خلال هذه الصورة

```
Loading DB routines from perl5db.pl version 1.28
Editor support available.

Enter h or `h h' for help, or `man perldebug' for more help.

Debugged program terminated.  Use q to quit or R to restart,
  use o inhibit_exit to avoid stopping after program termination
,
  h q, h R or h o to get additional info.
  DB<1>
```

FIGURE(31)

هذه هي احدى طرق استدعاء طرق المساعدة الخاصة بمصحح الاخطاء
مثل استدعاء صفحات ال
manual
وايضا لديك الايعاز

*C0D3

```
h
```

لعرض المساعدة وايضا لديك الايعاز

*C0D3

```
h h
```

واذا كان هذا الايعاز ايضا غير مفهوم يعطيك خيار اخر هو استعمال الامر

*C0D3

```
|h h
```

*N0T3

على اغلب الاحيان هذه الاوامر تعمل على اعطاء نفس النتائج و
المخرجات ولكن بصيغ مختلف بعض الشئ

وقفة مهمة في ما يتعلق بالموديلات الخاصة بلغة البيرل
كما ذكرنا انه الموديلات التي تعلمنا عملها في الجزء الاول من الكتاب اتفقنا على ان تتم تسميتها بالموديلات القياسية ولكن اذا تعمقت في لغة البيرل سوف تدرك ان ليس كل الموديلات التي ترغب بالعمل عليها تكون ملحقة بالمفسر البيرل لذا كان لا بد من حل موجود لكي يعمل على حل هذا النوع من المشاكل لذا فان الانترنت يوفر لنا حلا يكون موجود في موقع ال

www.cpan.org

وهذا الموقع مقسم الى اقسام اختر قسم الموديلات ثم البحث ومن ثم اعمل البحث عن الموديل الذي تريد
الان سوف افرض انك وجدت الموديل الذي ترغب بالبحث عنه فكيف سوف تنصبه على حاسبتك اولا يكون الموديل مكبوس تعمل على فك الكبس ثم تكتب هذه الخطوات بالتوالي ومع مراعاة حالة الاحرف كما في هذه الخطوات

1.perl Makefile.PL
2.make
3.make install

الان وعند انتهاء هذه الخطوات الثلاثة سوف يتم تنصيب الموديل بشكل كامل على مفسرك الخاص بلغة البيرل

*N0T3
now my magazine just done under the license of gpl
which allows you to make every think you ever want without an advance permission from me

GR33Tz:-

to every one help me out to make this magazine see the light and finally thanks went to every one in
www.programming-fr34ks.net
our new programming site coz they are such really nice friends and i wish the best luck for them

our special great went to our place of security our geeks home

www.securitygurus.net

we all miss and want you to come back again the real way of security

last touch

god *thx for helping me to do this work*

Mailme:: **mahmoud_najafy@hotmail.com**

*WrOT3By:-**M_SpAwN***